

# Toggling a Genetic Switch Using Reinforcement Learning

Aivar Sootla, Natalja Strelkowa, Damien Ernst, Mauricio Barahona, Guy-Bart Stan

**Abstract**—In this paper, we consider the problem of optimal exogenous control of gene regulatory networks. Our approach consists in adapting and further developing an established reinforcement learning algorithm called the fitted Q iteration. This algorithm infers the control law directly from the measurements of the system's response to external control inputs without the use of a mathematical model of the system. The measurement data set can either be collected from wet-lab experiments or artificially created by computer simulations of dynamical models of the system. The algorithm is applicable to a wide range of biological systems due to its ability to deal with nonlinear and stochastic system dynamics. To illustrate the application of the algorithm to a gene regulatory network, the regulation of the toggle switch system is considered. The control objective of this problem is to drive the concentrations of two specific proteins to a target region in the state space. In our companion paper, we take a closer look at the reference tracking problem from the reinforcement learning point of view and consider the generalised repressilator as an example.

**Index Terms**—Reinforcement learning, synthetic biology, fitted Q iteration, regulation, gene regulatory networks

## I. INTRODUCTION

Synthetic biology aims at the (re-)design of biological functions in living organisms for their use in various applications such as bioengineering, bioremediation and energy [1]. This is typically realised via the insertion of foreign genes inside a host cell (e.g., a bacterium *E. coli*). The expression of the foreign genes inside the host cells imposes *de facto* a burden on the native processes of the host cells. A high burden induces severe intracellular perturbations and can decrease cellular growth rate. This in turn disrupts the intended behaviour of synthetic biology gene networks [2]. Hence, it is highly desirable to develop means for controlling gene networks so as to efficiently enable the designed behaviour while simultaneously minimising the burden induced by this behaviour on the host cells.

The current biotechnology state-of-the-art allows us to quantitatively measure and interact with gene regulatory networks. Quantitative *in vivo* estimates of gene networks' states (outputs) can be obtained via fluorescent markers [3], [4] (e.g., green fluorescent protein, GFP or red fluorescent protein,

Aivar Sootla, Guy-Bart Stan (corresponding author) are with the Centre for Synthetic Biology and Innovation and the Department of Bioengineering, Mauricio Barahona is with the Department of Mathematics, Imperial College London, UK {a.sootla, g.stan, m.barahona}@imperial.ac.uk, Natalja Strelkowa is with Boehringer-Ingelheim Pharma GmbH & Co KG., Germany, natalja.strelkowa@boehringer-ingelheim.com, and Damien Ernst is with the Montefiore Institute, University of Liège, Belgium dernst@ulg.ac.be

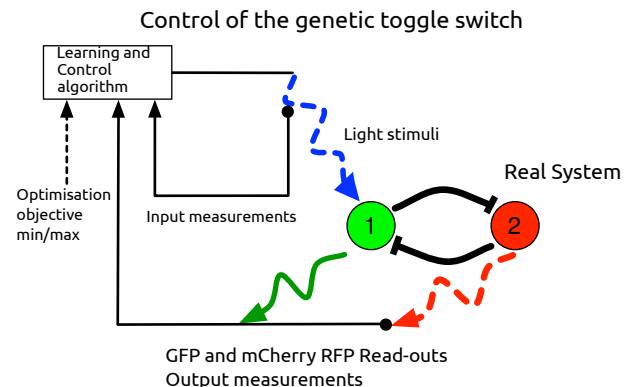


Fig. 1. A schematic depiction of the exogenously controlled genetic toggle switch. The green circle represents the *LacI* gene and the red circle represents the *TetR* gene. The arrows with flat ends represent repression of one gene by another. In the steady-state only one of the genes can be upregulated (or switched on). The goal is to toggle one of the genes, i.e., drive this gene from its downregulated mode to its upregulated one.

mCherry). A typical input is a targeted induction of the gene expression, which can be achieved by, e.g., conditional gene knock outs [5], [6], heat shocks [7] or monochromatic light pulses [8], [9]. This means that feedback control is technologically feasible *in vivo*. The objective of the control method can be minimal time control (i.e., driving the system as fast as possible to a target region in the state-space), minimal burden control (minimal expression of heterologous proteins), or a trade-off between the two, as considered in this paper. The control method must reach the objective, while maintaining the designed functions of a synthetic gene regulatory network.

Some control problems in gene regulatory networks were successfully addressed [10], [11], [12]. In all those papers, the authors used classical control methods, which infer the control law (or the control policy) based on a mathematical model of the system. One of the bottlenecks of these approaches is the modelling part, which for large gene regulatory networks is an extremely hard and lengthy process. Moreover, there are other challenges such as stochasticity. Stochasticity is expressed in the form of the intrinsic and extrinsic noise during gene expression [13]. Transcription and translation processes typically involve a few randomly interacting molecules, thus adding thermodynamic stochasticity to biochemical interactions.

The problems with modelling and stochasticity point towards the use of reinforcement learning methods [14], [15], which infer the control policy based solely on interactions with the real system. These methods do not require a physical model. Moreover, very few assumptions on the structure of the controlled system are made. However, the major advantage of

the reinforcement learning methods is to some extent their drawback. Indeed, these methods require interactions with the real system, which implies numerous costly and lengthy wet-lab experiments. A solution would be a reinforcement learning method, which learns the policy using a single experiment. For systems relevant to this paper, however, such a method will not be efficient. Indeed, a control policy, which tries to learn and control such systems in a single experiment, is generally not better than a random control policy [16]. In order to address these concerns, a hybrid approach is proposed. First, an initial control policy is computed using past experimental data and/or a mathematical model. After that the control policy is updated during the experiment using reinforcement learning methods. This approach will be applied to the regulation of the toggle switch system schematically depicted in Figure 1. The control objective of this problem is to drive the concentrations of two specific proteins to a target set in the state space and remain in this set.

The initial policy is obtained by the Fitted  $Q$  Iteration algorithm [17]. The algorithm requires only one-step system transitions to infer the control policy. A one-step system transition is a triplet  $\{\mathbf{n}, \mathbf{u}, \mathbf{n}^+\}$ , where  $\mathbf{n}^+$  denotes a successor state of the system in state  $\mathbf{n}$  subjected to input  $\mathbf{u}$ . Fitted  $Q$  Iteration can also handle nonlinear and stochastic systems and it is sample efficient. One-step transitions can be obtained by simulating the mathematical model of the system or using past experimental data. Afterwards the policy is updated by mixing the online measurements with past observations. The Exploration/Exploitation trade-off is addressed using an  $\varepsilon$ -greedy policy.

This paper is organised as follows. Mathematical preliminaries are described in Section II. In order to make the paper self contained, the fitted  $Q$  algorithm is sketched and different aspects of modelling in gene regulatory networks are discussed. The problem of controlling the toggle switch is formulated and discussed in detail in Section III. The algorithm is described in Section IV; finally, the simulation results are presented in Section V. Reference trajectory tracking for the generalised repressilator system is the subject of our companion paper [18].

## II. PRELIMINARIES

### A. Modelling in Biology

At the cellular level chemical reactions depend on thermodynamical principles, since molecules must collide before a reaction can start. Therefore chemical reactions inside living organisms are modelled using stochastic calculus. The following approach to chemical reaction modelling is described in detail in [19]. Consider a well-stirred system of  $k$  species in a constant volume and a thermal equilibrium. Assume the species are interacting through  $m$  reactions. Let  $N^i(t)$  be the number of molecules of species  $i$  and  $\nu_{ij}(t)$  be the change in the molecular concentration of species  $i$  at time  $t$  if the reaction  $j$  occurs. The bold symbols will be used to denote vectors, e.g.,  $\mathbf{N}$  stands for the vector with elements  $N^i$ . Finally, let  $a_j(\mathbf{n})dt$  be the probability of reaction  $j$  occurring in the next infinitesimal interval  $[t, t + dt]$ , if the number of molecules at

time  $t$ ,  $\mathbf{N}(t)$ , is equal to  $\mathbf{n}$ . The functions  $a_j(\cdot)$  are called propensity functions. The time evolution of the concentration of species can be modelled by a Markov stochastic process, for which:

$$\frac{\partial \Pr(\mathbf{n}, t | \mathbf{n}_0, t_0)}{\partial t} = \sum_{j=1}^m a_j(\mathbf{n} - \boldsymbol{\nu}_j) \Pr(\mathbf{n} - \boldsymbol{\nu}_j, t | \mathbf{n}_0, t_0) - a_j(\mathbf{n}) \Pr(\mathbf{n}, t | \mathbf{n}_0, t_0) \quad (1)$$

where the probability  $\Pr(\mathbf{n}, t | \mathbf{n}_0, t_0)$  stands for  $\Pr(\mathbf{N}(t) = \mathbf{n} | \mathbf{N}(t_0) = \mathbf{n}_0)$ . This equation is called the *chemical master equation*. Instead of computing particular realisations, one can also obtain the expression for the mean. This can be done by multiplying (1) with  $\mathbf{n}$ , summing over all  $\mathbf{n}$  and using  $\langle h(\mathbf{N}(t)) \rangle = \sum_{\mathbf{n}} h(\mathbf{n}) \Pr(\mathbf{n}, t | \mathbf{n}_0, t_0)$ , where  $\langle \cdot \rangle$  stands for the mean.

$$\frac{d\langle \mathbf{N}(t) \rangle}{dt} = \sum_{j=1}^m \boldsymbol{\nu}_j a_j(\langle \mathbf{N}(t) \rangle) \quad (2)$$

If by assumption  $\mathbf{N}(t)$  is a deterministic function then  $\langle \mathbf{N}(t) \rangle$  is equal to  $\mathbf{N}(t)$ . Moreover, equation (2) describes the species' concentrations in the confined volume where the chemical reactions take place.

### B. Formulation of the Optimal Control Problem

Consider a deterministic discrete-time dynamical system

$$\mathbf{n}_{t+1} = f(\mathbf{n}_t, \mathbf{u}_t) \quad (3)$$

where  $\mathbf{u}_t$  is the control input at time  $t$ , which belongs to a compact set  $U$  for every  $t$ . In the stochastic case, Markov decision processes (MDPs) are typically employed, for which

$$\Pr(\mathbf{n}_{t+1} \in \mathbf{N}_{t+1} | \{\mathbf{n}_k\}_{k=0}^t, \{\mathbf{u}_k\}_{k=0}^t) = \Pr(\mathbf{n}_{t+1} \in \mathbf{N}_{t+1} | \mathbf{n}_t, \mathbf{u}_t).$$

The above relationship means that the probability of the state  $\mathbf{n}_{t+1}$  belonging to the set  $\mathbf{N}_{t+1}$  does not depend on the entire history of the realisation of the states  $\{\mathbf{n}_k\}_{k=0}^t$  and control signals  $\{\mathbf{u}_k\}_{k=0}^t$ , but depends only on the current values  $\mathbf{n}_t$  and  $\mathbf{u}_t$ . Under the above Markovian assumption, dynamical stochastic systems can be modelled as

$$\Pr(\mathbf{n}_{t+1} \in \mathbf{N}_{t+1} | \mathbf{n}_t, \mathbf{u}_t) = \int_{\mathbf{N}_{t+1}} f(\mathbf{n}_t, \mathbf{u}_t, x) dx,$$

or in a compact form

$$\mathbf{n}_{t+1} \sim f(\mathbf{n}_t, \mathbf{u}_t, \cdot).$$

Here, we slightly abuse the notation by using again the symbol  $f$  as in the deterministic system (3). This is done, in order to signify that these functions describe the dynamics of the system whether it is stochastic or deterministic.

In both cases, consider an optimal control problem, which is defined through the minimisation of an infinite sum of discounted costs  $\mathbf{c}(\mathbf{n}, \mathbf{u})$ . In the deterministic case the problem is defined as

$$V(\mathbf{n}_t) = \min_{\mu(\cdot): \mu(\mathbf{n}_i) = \mathbf{u}_i} \sum_{i=t}^{\infty} \gamma^{i-t} \mathbf{c}(\mathbf{n}_i, \mathbf{u}_i)$$

and in the stochastic case as

$$V(\mathbf{n}_t) = \min_{\mu(\cdot): \mu(\mathbf{n}_i) = \mathbf{u}_i} \lim_{K \rightarrow \infty} \mathbb{E}_{\mathbf{n}_{t+1} \sim f(\mathbf{n}_t, \mathbf{u}_t, \cdot)} \sum_{i=t}^K \gamma^{i-t} c(\mathbf{n}_t, \mathbf{u}_t)$$

where  $V(\mathbf{n}_t)$  is called the value function and  $\mu(\cdot)$  is a mapping from  $\mathbf{n}$  to  $\mathbf{u}$ , which is called the control policy. The cost function  $c$  specifies the objective of the control problem, which in our case is driving the system to a specific region in the state-space. In our setting, the control policy should be inferred based only on realisations of one-step system transitions  $\{\mathbf{n}_l, \mathbf{u}_l, \mathbf{n}_l^+\}$ , where  $\mathbf{n}_l^+$  is a successor state of the system in the state  $\mathbf{n}_l$  and subjected to the input  $\mathbf{u}_l$  (in the deterministic case, if the function  $f(\cdot, \cdot)$  is known  $\mathbf{n}_l^+$  is equal to  $f(\mathbf{n}_l, \mathbf{u}_l)$ ). For the purpose of this paper, the function  $c(\cdot, \cdot)$  is assumed to be known in advance.

### C. Fitted Q Iteration

A central object of the fitted  $Q$  algorithm is the  $Q$  function, which is introduced as follows:

$$Q(\mathbf{n}_t, \mathbf{u}_t) = c(\mathbf{n}_t, \mathbf{u}_t) + \min_{\mu(\cdot)} \sum_{i=t}^{\infty} \gamma^{i-t} c(\mathbf{n}_i, \mu(\mathbf{n}_i))$$

Once a  $Q$  function is computed, the optimal feedback control policy is given as:

$$\mu^*(\mathbf{n}) = \operatorname{argmin}_{\mathbf{u} \in U} Q(\mathbf{n}, \mathbf{u})$$

Under certain conditions (which form the celebrated optimality principle), the  $Q$  function can be obtained as the unique solution of the following iterative procedure:

$$Q_k(\mathbf{n}, \mathbf{u}) = c(\mathbf{n}, \mathbf{u}) + \gamma \min_{\mathbf{u}' \in U} Q_{k-1}(f(\mathbf{n}, \mathbf{u}), \mathbf{u}') \quad (4)$$

where  $Q_0$  is equal to  $c$ . However, (4) is hard to solve in general, especially if only the triplets  $\mathcal{F}$  are given. Therefore an approximation  $\hat{Q}$  of the  $Q$  function is computed using an iterative procedure. Let  $\hat{Q}_0 = c$  and for every  $(\mathbf{n}_l, \mathbf{u}_l, \mathbf{n}_l^+)$  in  $\mathcal{F}$  compute:

$$\hat{Q}_1(\mathbf{n}_l, \mathbf{u}_l) = c(\mathbf{n}_l, \mathbf{u}_l) + \gamma \min_{\mathbf{u} \in U} \hat{Q}_0(\mathbf{n}_l^+, \mathbf{u})$$

This expression gives  $\hat{Q}_1$  only for  $\mathbf{n}_l, \mathbf{u}_l$  in  $\mathcal{F}$ , while the entire function  $\hat{Q}_1(\cdot, \cdot)$  is estimated by a regression algorithm (e.g., EXTRA Trees [20]). This can be generalised to an iterative procedure, which can be used to obtain a near-optimal control policy as outlined in Algorithm 1. The stopping criterion can be simply the maximum number of iterations  $N_{\text{it}}$ , which is chosen such that the number  $\gamma^{N_{\text{it}}}$  is sufficiently small and the values  $\hat{Q}_k(\mathbf{n}_l, \mathbf{u}_l)$  are not modified significantly for  $k$  larger than  $N_{\text{it}}$ . Other criteria are described in [17]. Due to space limitation a more detailed method description is given in our companion paper [18]. Note that Algorithm 1 can be extended to handle the stochastic case as well [17].

---

### Algorithm 1 Fitted $Q$ iteration algorithm

---

**Inputs:** Set of triplets  $\mathcal{F} = \{\mathbf{n}_l, \mathbf{u}_l, \mathbf{n}_l^+\}_{l=1}^{\#\mathcal{F}}$ , stopping criterion, cost function  $c(\cdot, \cdot)$

**Outputs:** Policy  $\hat{\mu}^*(\mathbf{n})$

$k \leftarrow 0$

$\hat{Q}_0(\cdot, \cdot) \leftarrow c(\cdot, \cdot)$

**repeat**

$k \leftarrow k + 1$

In order to obtain the values of  $\hat{Q}_k(\cdot, \cdot)$  for all  $\{\mathbf{n}_l, \mathbf{u}_l\}$  in  $\mathcal{F}$  compute:

$$\hat{Q}_k(\mathbf{n}_l, \mathbf{u}_l) = c(\mathbf{n}_l, \mathbf{u}_l) + \gamma \min_{\mathbf{u} \in U} \hat{Q}_{k-1}(\mathbf{n}_l^+, \mathbf{u}) \quad (5)$$

Estimate the function  $\hat{Q}_k(\mathbf{n}, \mathbf{u})$  using a regression algorithm with input pairs  $(\mathbf{n}_l, \mathbf{u}_l)$  and function values  $\hat{Q}_k(\mathbf{n}_l, \mathbf{u}_l)$ .

**until** the stopping criterion is satisfied

Compute  $\hat{\mu}^*(\mathbf{n}) = \operatorname{argmin}_{\mathbf{u} \in U} \hat{Q}_k(\mathbf{n}, \mathbf{u})$

---

## III. SYSTEM DESCRIPTION AND PROBLEM SETTING

The genetic toggle switch introduced in [21] consists of the *LacI* and *TetR* genes mutually repressing each other (see Figure 1). The mutual repression means that the increase in the protein product concentration of one gene implies the decrease in the protein product concentration of the other gene. It can be shown that this system typically has two stable fixed points. At each of these stable fixed points, one gene is downregulated (switched off), while the other gene is upregulated (switched on). This means that only one of two genes can be switched “on” at the steady state. In the sequel, we will use the following numeric references. Gene 1 and protein 1 will denote the *LacI* gene and the protein produced by its expression, respectively. Similarly, gene 2 and protein 2 will denote the *TetR* gene and its expression product, respectively. We assume that for both genes the protein concentrations are given as readouts, for instance via fluorescent markers. On the other hand, we assume that the control inputs are implemented as light pulses activating a photo-sensitive promoter controlling the expression of gene 1. When this photo-sensitive promoter is activated through a light pulse the concentration of the gene product 1 is increased by a small amount through the expression of gene 1. Basic mass-action kinetics of the toggle switch result in a high-order model, which is typically reduced to a two state model using time scale separation (cf. [22]). This can be done because most of the reactions occur on a fast time scale (order of seconds) in comparison with the gene expression time scale (order of minutes or even hours). The fast time scale includes the mRNA dynamics and the light-induction of the promoter [8]. We approximate the action of light induction  $u_t$  as a discrete variable in the set  $U = \{0, 1\}$ . The reduced order model has two states, which are the two

protein concentrations:

$$\begin{aligned} n_{t+1}^1 &= \frac{c_1}{1 + (n_t^2)^{\alpha_2}} - d_1 n_t^1 + bu_t \\ n_{t+1}^2 &= \frac{c_2}{1 + (n_t^1)^{\alpha_1}} - d_2 n_t^2 \end{aligned} \quad (6)$$

where  $n_t^i$  is the concentration of the protein  $i$  at time  $t$ ,  $c_i$  is the effective rate of synthesis of the repressor  $i$ ,  $\alpha_i$  is the cooperativity coefficient of the repressor  $i$ ,  $d_i$  is the degradation rate of protein  $i$ , and  $b$  is the number of proteins produced per unit of time as a result of one light pulse. A more realistic model would also have a time-delayed control action and a “leaky” transcription on both gene expressions. Such extensions require simple modifications of our control algorithm, but make the results less transparent and harder to analyse.

The stochastic model, which is also used in the sequel, is derived based on the deterministic model (6), i.e., the propensities are obtained from the reduced order model [23]. This results in the following Markov decision process:

$$\begin{aligned} \frac{\partial \Pr(n^1, n^2, t)}{\partial t} &= (d_2(n^2 - 1) + g_2) \Pr(n^1, n^2 - 1, t) + \\ &\quad (g_1 + bu_t + d_1(n^1 - 1)) \Pr(n^1 - 1, n^2, t) - \\ &\quad (d_1 n^1 + d_2 n^2 + g_1 + g_2 + bu_t) \Pr(n^1, n^2, t) \end{aligned} \quad (7)$$

where

$$g_1 = \frac{c_1}{1 + (n_t^2)^{\alpha_2}} \text{ and } g_2 = \frac{c_2}{1 + (n_t^1)^{\alpha_1}}$$

The control objective in both deterministic and stochastic settings is to start from any initial condition in the vicinity of the first fixed point (for which gene 1 is off) and drive the system to the vicinity of the second fixed point (for which gene 1 is on) by applying appropriate control signals to gene 1. At the same time the optimal control objective needs to address the trade-off between minimum time control and minimum burden control.

#### IV. CONTROL ALGORITHM

Our goal is to develop a control algorithm, which learns how to near-optimally control the toggle switch system in a single experiment. Toggling the switch can be done experimentally in a couple of hours and the fastest measurement sampling is in the order of one minute. This gives at most 200 samples in a single trajectory. Learning a near-optimal control policy for toggling the switch with such limited amount of data is an extremely hard problem to solve. To tackle this issue, we propose to first learn a “rough approximation” of the control policy obtained by applying Algorithm 1 to one-step system transitions  $\mathcal{F}$  artificially generated from simulations of a mathematical model. Afterwards the policy is fine-tuned by mixing the online measurements with past observations  $\mathcal{F}$ . The Exploration/Exploitation trade-off is addressed using an  $\varepsilon$ -greedy policy.

Our approach is outlined in Algorithm 2. Let  $\hat{Q}_{\text{Alg 1}}(\cdot, \cdot)$  be the approximation of the  $Q$  function obtained by Algorithm 1. While experiment is running new input-output samples are collected in  $\mathcal{F}_{\text{new}}$  through direct interaction with the real

---

#### Algorithm 2 Online learning algorithm

---

**Inputs:** Set  $\mathcal{F} = \{\mathbf{n}_l, \mathbf{u}_l, \mathbf{n}_l^+\}_{l=1}^{\#\mathcal{F}}$ , cost function  $c(\cdot, \cdot)$ , the function  $\hat{Q}_{\text{Alg 1}}(\cdot, \cdot)$ , number of iterations  $N$ , function  $h(\cdot, \cdot)$

$\tilde{Q}_0(\cdot, \cdot) \leftarrow \hat{Q}_{\text{Alg 1}}(\cdot, \cdot)$

$\mathcal{F}_{\text{cur}} \leftarrow \mathcal{F}$

**while** new data is received **do**

$k \leftarrow 0$

Collect a set of new samples  $\mathcal{F}_{\text{new}} =$

$\{\mathbf{n}_m, \mathbf{u}_m, \mathbf{n}_m^+\}_{m=1}^{\#\mathcal{F}_{\text{new}}}$

$\mathcal{F}_{\text{cur}} \leftarrow h(\mathcal{F}_{\text{cur}}, \mathcal{F}_{\text{new}})$

**while**  $k \leq N$  **do**

$k \leftarrow k + 1$

In order to obtain the values of  $\tilde{Q}_k(\cdot, \cdot)$  for all  $\{\mathbf{n}_l, \mathbf{u}_l\}$  in  $\mathcal{F}_{\text{cur}}$  compute:

$$\tilde{Q}_k(\mathbf{n}, \mathbf{u}) = c(\mathbf{n}, \mathbf{u}) + \gamma \min_{\mathbf{u}' \in U} \tilde{Q}_{k-1}(\mathbf{n}^+, \mathbf{u}') \quad (8)$$

Estimate the function  $\tilde{Q}_k(\mathbf{n}, \mathbf{u})$  using a regression algorithm with input pairs  $(\mathbf{n}_l, \mathbf{u}_l)$  and function values  $\tilde{Q}_k(\mathbf{n}_l, \mathbf{u}_l)$ .

**end while**

$\tilde{Q}_0(\cdot, \cdot) \leftarrow \tilde{Q}_k(\cdot, \cdot)$

**end while**

---

system. Then the approximation of the  $Q$  function is updated as prescribed in the inner loop of Algorithm 2. After that the new set  $\mathcal{F}_{\text{new}}$  is formed and new samples are collected.

The major challenge of Algorithm 2 is appropriately choosing the function  $h(\cdot, \cdot)$ , which combines the sets  $\mathcal{F}_{\text{cur}}$  and  $\mathcal{F}_{\text{new}}$ . As an example, we consider  $h(\mathcal{F}_{\text{cur}}, \mathcal{F}_{\text{new}}) = \mathcal{F}_{\text{cur}} \cup \mathcal{F}_{\text{new}}$ . Such a choice has some drawbacks. If the initial set  $\mathcal{F}$  contains many samples, then the updates in (8) will not result in significant changes in the policy. This happens because the algorithm appreciates equally the samples in  $\mathcal{F}$  and the new sets  $\mathcal{F}_{\text{new}}$ , even though the samples in  $\mathcal{F}$  are artificially generated using a mathematical model and the samples in  $\mathcal{F}_{\text{new}}$  are obtained from the real system.

The choice of other parameters is guided by the following considerations. The number of iterations  $N$  is chosen according to the computational constraints in the control system. In our simulations, we consider the worst case scenario and assume that  $N$  is equal to one.

An important task of such a learning algorithm is a trade-off between exploration and exploitation in generation of the set  $\mathcal{F}_{\text{new}}$ . The exploration is required, since the real system is essentially unknown to the algorithm and the exploratory actions will provide new information. The trade-off policy between exploration and exploitation is defined as follows:

$$\mathbf{u}_t = \begin{cases} \text{argmin}_{\mathbf{u}' \in U} \tilde{Q}_k(\mathbf{n}_t, \mathbf{u}') & \text{with probability } \varepsilon_t \\ \text{random action} & \text{with probability } 1 - \varepsilon_t \end{cases}$$

where  $\mathbf{n}_t$  is the state measured at time  $t$  and  $\tilde{Q}_k(\cdot, \cdot)$  is a current approximation of the  $Q$  function. In our experiment  $\varepsilon_t$  is an increasing function of  $t$  between zero and one. During the

first time samples, the need for new information is typically higher, and thus a low value of  $\varepsilon_t$  should be chosen.

The structure of the instantaneous cost  $c(\mathbf{n}, u)$  is chosen as follows:

$$c(n^1, n^2, u) = n^1 + \alpha_2 n^2 + \alpha_u u$$

where  $\alpha_2, \alpha_u$  are non-negative constants. This choice of cost function is based on the fact that, in most biological systems, both  $\mathbf{n}$  and  $u$  take only non-negative values. Linear cost functions are very useful in such case as was shown in [24] for linear time-invariant systems with non-negative states and inputs. The weight  $\alpha_2$  influences the switching time; the larger the value of  $\alpha_2$  the faster the switch. Since our goal is to toggle the genetic switch depicted in Figure 1 from the situation, where  $n^1$  is small and  $n^2$  is large to the opposite situation, the constant  $\alpha_2$  must be larger than one. Our target region in the state-space is a neighbourhood around the steady-state, where  $n^1$  is large and  $n^2$  is small. Therefore, this cost will ensure that the target region is reached without specifying this region explicitly in the cost. Finally, the term  $\alpha_u u$  penalises the control signal and therefore attempts to minimise the burden associated with light-induced gene expression. The choice of the weights  $\alpha_2$  and  $\alpha_u$  dictates the trade-off that exists between toggling the switch fast and toggling the switch with a reduced gene expression burden.

In the stochastic case, the direct Gillespie stochastic simulation algorithm [19] is used to compute the value  $\mathbf{n}_{t+1}$  based on  $\mathbf{n}_t$ . Each trajectory of the algorithm starts at  $\mathbf{n}_t$  and is computed until the next sampling time  $t + 1$ . The value  $\mathbf{n}_{t+1}$  is then averaged over one hundred trajectories. In terms of Algorithm 1, the only difference with the deterministic case is in the choice of the parameter  $n_{min}$ . This parameter prunes the regression trees and makes the  $\hat{Q}$  function “smoother”. The choice of this parameter is discussed in the next section.

The algorithm is implemented in Python using the machine learning [25], parallelisation [26], graphics [27] and scientific computation [28] toolboxes.

## V. SIMULATION PARAMETERS AND RESULTS

As an illustration of the benefits of our proposed approach, we investigate how our algorithm handles model uncertainty. For this, we investigate how a control policy learned in batch-mode from data collected from one system can be adapted in online mode using data collected from a second system to control this latter system. Consider the system (6) with parameters  $d_1 = 1, d_2 = 1, b = 20$ . The rest of the parameters are chosen according to two settings:

$$\alpha_1 = 1 \quad \alpha_2 = 3 \quad c_1 = 30 \quad c_2 = 10 \quad (9)$$

$$\alpha_1 = 3 \quad \alpha_2 = 1 \quad c_1 = 40 \quad c_2 = 60 \quad (10)$$

These parameters are almost “opposed” in terms of the corresponding fixed point values of gene 1 and gene 2, making these two settings very different, even though the systems are structurally similar. In order to compute the initial  $Q$  function, we generate 5000 trajectories with 50 samples in each trajectory. The discount factor  $\gamma$  is chosen equal to 0.75. This parameter represents the trade-off between fast convergence of the algorithm and the importance attributed

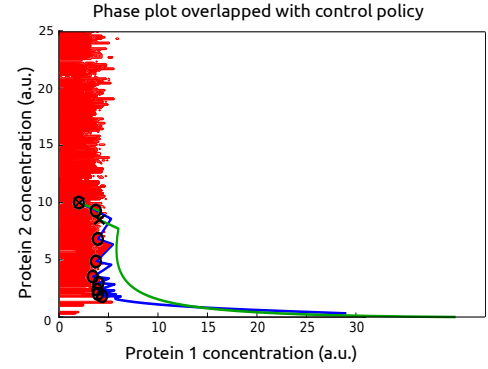


Fig. 2. Control of a deterministic model of the genetic toggle switch. The phase plot of the control policy is overlapped with the trajectories; the red area marks the region of the state space for which the near-optimal control action is equal to one, while the white area marks the region of the state space for which the near-optimal control policy is here learned from the system with parameter values given in (9). Circles and crosses denote the application of the control action. The blue trajectory correspond to the system with parameter values given in (9), while the green trajectory corresponds to the system with parameter values given in (10).

to future control actions. Note that the closer  $\gamma$  is to 1, the more importance is attributed to future control actions. The weights in the cost function are chosen such that  $\alpha_2$  is equal to 60 and  $\alpha_u$  is equal to 1.

First, let us consider the deterministic case. The control policy is learned from the system with parameter values given in (9) and applied to both systems with parameter values in (9) and (10). As can be seen in Figure 2 both switches are toggled successfully. Even when the system with parameter values given in (10) is controlled the algorithm is able to fulfil the control objective. In order to comment further on the results, we can take a closer look at the computed control policy depicted in Figure 2. Essentially, a threshold for protein 1 is set such that, if its concentration falls below a certain value, then a control action is applied to drive up its concentration. Eventually, protein 1 becomes dominant and its concentration starts to increase due to the dynamics of the toggle switch system. At the same time, the concentration of protein 2 starts to decrease. The threshold value (i.e., the boundary between the red and white regions in Figure 2) can be adjusted through the value of the weights appearing in the definition of the instantaneous cost function. For example, if the weight on the control signal  $u$  is increased (respectively decreased), the threshold moves to the left (respectively to the right).

The simulation results in Figure 2 might seem impressive. However, such behaviour of the controlled system is not always achievable. In our second set of simulations, the threshold value is not sufficient to drive the system in the region of attraction of the second equilibrium (see upper panel of Figure 3). To address this problem, we use Algorithm 2 as described in the previous section. Figure 3 illustrates that Algorithm 2 can serve as an adaptation technique that compensates for model uncertainty and variations in the controlled system. Algorithm 2 is able to toggle the switch in the setting which proved to be difficult to handle for the purely batch-mode algorithm.

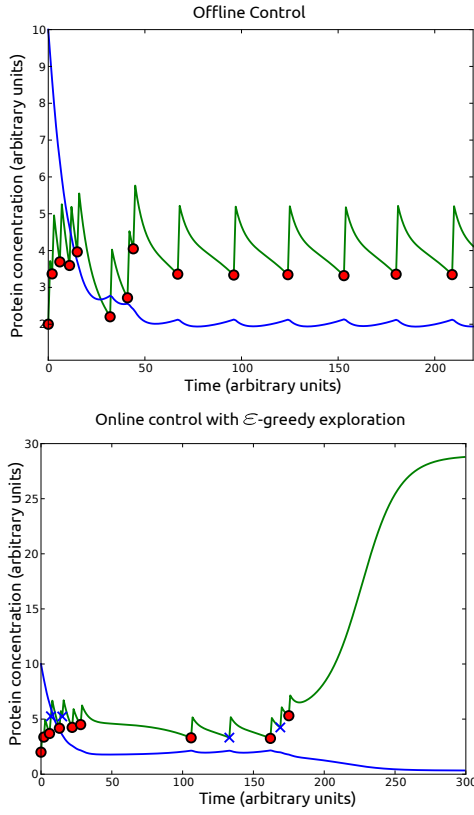


Fig. 3. Trajectories of the protein dynamics in the offline (batch-mode) and the online fitted Q iteration modes. In the upper panel we can see that the control policy inferred from input-output data of the system with parameter values given in (10) is not able to adequately toggle the genetic switch with parameter values given in (9). The system does not converge to its the desired state and oscillations are produced instead. In the lower panel, we start with the same policy. However, the policy is now adjusted using Algorithm 2 applied to newly obtained data. The red circles in both panels represent the control action obtained from the available control policy. The blue crosses are the exploratory actions, which are applied according to the  $\varepsilon$ -greedy strategy. This approach manages to provide the algorithm with a sufficient amount of information in order to solve the optimal control problem of efficiently toggling the switch.

The stochastic case simulation setting is very similar to the deterministic one. For the stochastic case, the parameter  $n_{min}$  is chosen equal to 5, which provides a control policy similar to the deterministic case depicted in Figure 2. In general, the parameter  $n_{min}$  can be chosen automatically using cross-validation methods (cf. [29]). The simulation results in the stochastic setting are depicted in Figure 4. The simulation is performed using direct Gillespie stochastic simulation algorithm. At every time instance  $t$ , one hundred trajectories starting at  $\mathbf{n}_t$  are computed until the next time instance  $t + 1$ , and the value  $\mathbf{n}_{t+1}$  is then averaged over these trajectories. The stochastic simulation setting is similar to the deterministic one depicted in Figure 3. Unlike the simulation in Figure 3, for this stochastic simulation the toggle is switched; however, this is just one realisation. On average the situation will be similar to the deterministic case.

## VI. DISCUSSION AND CONCLUSION

A framework for data-based inference of feedback control laws is presented. We show that this framework can efficiently

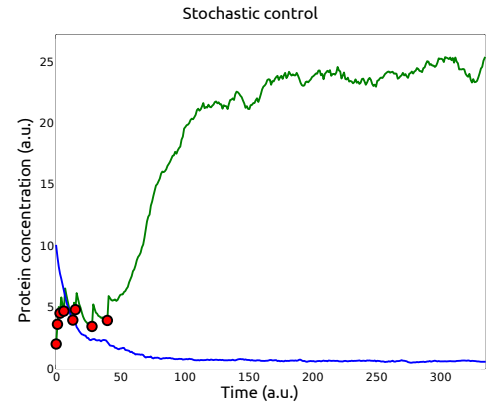


Fig. 4. Control of a stochastic model of the genetic toggle switch. The curves represent the average trajectories obtained when the control policy is learned from the system with parameter values given in (10) and then applied to the system with parameter values given in (9). The simulation is performed using direct Gillespie stochastic simulation algorithm. At every time step  $t$ , one hundred trajectories starting at  $\mathbf{n}_t$  are computed until the next time instance  $t + 1$ , and the value  $\mathbf{n}_{t+1}$  is then averaged over these trajectories.

handle the control of gene regulatory networks. The objective of the optimal control problem is chosen as a trade-off between minimum time control and minimum burden control.

The major feature of the presented control algorithm is its adaptive nature. The algorithm computes an initial control policy using a batch-mode reinforcement learning method. The input to the method are one-step transitions of the system, which can be obtained using the experimental data and/or simulation of a mathematical model. Then this policy is adapted to the real system based on newly obtained measurements. The algorithm, however, does not take into account the *a priori* knowledge that it will be applied to a different, but structurally similar system. A correct exploitation of structural similarity between the learned from and applied to systems may significantly improve the performance of the presented algorithm. This constitutes one of the main directions for future work that is currently under investigation.

## REFERENCES

- [1] Priscilla E M Purnick and Ron Weiss. The second wave of synthetic biology: from modules to systems. *Nature reviews Molecular cell biology*, 10(6):410–422, jun 2009.
- [2] Cheemeng Tan, Philippe Marguet, and Lingchong You. Emergent bistability by a growth-modulating positive feedback circuit. *Nature chemical biology*, 5(11):842–848, November 2009.
- [3] L. Cai, N. Friedman, and X. S. Xie. Stochastic protein expression in individual cells at the single molecule level. *Nature*, 440:358–362, 2006.
- [4] Matthew R. Bennett and Jeff Hasty. Microfluidic devices for measuring gene network dynamics in single cells. *Nat Rev Genet*, 10(9):628–638, September 2009.
- [5] Natalia Ivanova, Radu Dobrin, Rong Lu, Iulia Kutenko, John Levorso, Christina DeCoste, Xenia Schafer, Yi Lun, and Ihor R. Lemischka. Dissecting self-renewal in stem cells with rna interference. *Nature*, 442:533–538, 2006.
- [6] Yu Liu, Masanori Asakura, Hironori Inoue, Teruya Nakamura, Motoaki Sano, Zhiyong Niu, Michelle Chen, Robert J. Schwartz, and Michael D. Schneider. Sox17 is essential for the specification of cardiac mesoderm in embryonic stem cells. *Proc Natl Acad Sci USA*, 104(10):3859–3864, 2007. <http://www.pnas.org/content/104/10/3859.full.pdf+html>.
- [7] Jerome T. Mettetal, Dale Muzzey, Carlos Gomez-Urbe, and Alexander van Oudenaarden. The Frequency Dependence of Osmo-Adaptation in *Saccharomyces cerevisiae*. *Science*, 319(5862):482–484, 2008.



- [8] Sae Shimizu-Sato, Enamul Huq, James M. Tepperman, and Peter H. Quail. A light-switchable gene promoter system. *Nat Biotech*, 20(10):1041–1044, 2002.
- [9] Anselm Levskaya, Orion D. Weiner, Wendell A. Lim, and Christopher A. Voigt. Spatiotemporal control of cell signalling using a light-switchable protein interaction. *Nature*, 461:997–1001, 2009.
- [10] Filippo Menolascina, Mario Di Bernardo, and Diego Di Bernardo. Analysis, design and implementation of a novel scheme for in-vivo control of synthetic gene regulatory networks. *Automatica, Special Issue on Systems Biology*, 47(6):1265–1270, April 2011.
- [11] Jannis Uhlendorf, Agnès Miermont, Thierry Delaveau, Gilles Charvin, François Fages, Samuel Bottani, Gregory Batt, and Pascal Hersen. Long-term model predictive control of gene expression at the population and single-cell levels. *Proceedings of the National Academy of Sciences*, 109(35):14271–14276, 2012.
- [12] Andreas Miliadis-Argeitis, Sean Summers, Jacob Stewart-Ornstein, Ignacio Zuleta, David Pincus, Hana El-Samad, Mustafa Khammash, and John Lygeros. In silico feedback for in vivo regulation of a gene expression circuit. *Nature biotechnology*, 29:1114–1116, 2011.
- [13] Peter S. Swain, Michael B. Elowitz, and Eric D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proc Natl Acad Sci USA*, 99(20):12795–12800, 2002, <http://www.pnas.org/content/99/20/12795.full.pdf+html>.
- [14] R.S. Sutton and A.G. Barto. *Reinforcement Learning, an Introduction*. MIT Press, 1998.
- [15] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Pr I Llc, 2010.
- [16] Michael Castronovo, Francis Maes, Raphael Fonteneau, and Damien Ernst. Learning exploration/exploitation strategies for single trajectory reinforcement learning. In *Proceedings of the 10th European Workshop on Reinforcement Learning (EWRL 2012)*, 2012.
- [17] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, April 2005.
- [18] A. Sootla, N. Strelkowa, D. Ernst, M. Barahona, and G.B. Stan. On reference tracking using reinforcement learning with application to gene regulatory networks. In *submission to the 52nd Conf. on Decision and Control*, Dec. 2013, <http://www3.imperial.ac.uk/people/a.sootla/publications>.
- [19] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.
- [20] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.
- [21] Timothy Gardner, Charles R. Cantor, and James J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403:339–342, 2000.
- [22] Ra’ul Guantes and Juan F Poyatos. Dynamical principles of two-component genetic oscillators. *PLoS Comput Biol*, 2(3):e30, 2006.
- [23] M. Hemberg and M. Barahona. Perfect sampling of the master equation for gene regulatory networks. *Biophysical journal*, 93(2):401, 2007.
- [24] C. Briat. Robust stability and stabilization of uncertain linear positive systems via integral linear constraints:  $L_1$ -gain and  $L_\infty$ -gain characterization. *Int J of Rob & Nonl Cont*, 2012.
- [25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [26] Joblib: running python function as pipeline jobs. <http://packages.python.org/joblib>.
- [27] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.
- [28] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. 2001–, <http://www.scipy.org/>.
- [29] Mervyn Stone. Cross-validated choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–147, 1974.